Fondamenti di Informatica T-1, 2019/2020 – Modulo 2

Prova d'Esame 3A di Giovedì 13 Febbraio 2020 – tempo a disposizione 2h

Avvertenze per la consegna: apporre all'inizio di ogni file sorgente un commento contenente i propri dati (**cognome, nome, numero di matricola**) e il **numero** della prova d'esame. Al termine, **consegnare tutti i file sorgenti** necessari alla compilazione e alla corretta esecuzione del programma.
Nota: **il main non è opzionale**; i *test* richiesti vanno implementati.
Consiglio: per verificare l'assenza di *warning*, eseguire sempre *"Rebuild All"*.

Un importante costruttore bolognese sta ultimando la costruzione di un grande palazzo in centro città e tiene traccia delle trattative di vendita ed affitto utilizzando un sistema software dedicato. In un file di testo sono elencate tutte le unità immobiliari. Per ciascuna di esse su ogni riga sono memorizzate le seguenti informazioni tutte separate da uno spazio: l'**identificativo** univoco dell'unità (un intero), la **natura** residenziale 'R' o commerciale 'C' (un carattere, tipo char), la **superficie** in metri quadri (un float), il **piano** (un intero), il numero di **vani** (un intero, al massimo vi sono 5 vani) e **per ciascun vano** la superficie calpestabile (un float).
In un secondo file di testo si tiene traccia delle trattative in corso. Su ogni riga è specificato per primo l'identificativo di una unità immobiliare, poi nome e cognome del potenziale acquirente fra doppi apici (un'unica stringa di al più 2047 caratteri utili, contenente spazi) ed in fine il valore dell'offerta in € (un float).
Si vedano, a titolo di esempio, i file "`palazzo.txt`" e "`offerte.txt`" forniti con lo StartKit.

Esercizio 1 – Struttura dati Appartamento, e funzioni di lett./scritt. (mod. element.h e palazzo.h/.c)

Si definisca una struttura dati `Appartamento` per memorizzare i dati relativi ad una unità immobiliare come sopra descritta.
Si definisca la funzione:

```
Appartamento leggiUnAppartamento(FILE * fp);
```

che, ricevuto in ingresso un puntatore ad una struttura dati di tipo `FILE`, facente riferimento ad un file con l'elenco degli appartamenti, legga i dati contenuti su una sola riga e li restituisca tramite una struttura dati di tipo `Appartamento`. Qualora vi siano problemi nella lettura, la funzione restituisca una struttura dati con identificativo dell'appartamento pari a -1.

Si definisca la funzione:

```
list leggiTuttiAppartamenti(char * fileName);
```

che, ricevuto in ingresso il nome di un file che contenga l'elenco degli appartamenti, legga da tale file i dati di tutti gli appartamenti e li restituisca in una lista di strutture dati di tipo Appartamento. Qualora la funzione incontri dei problemi nella lettura, o vi siano errori nell'apertura del file, la funzione dovrà stampare un messaggio di errore a video, e restituire una lista vuota.

Il candidato abbia cura di realizzare nel main opportuni test al fine di verificare il corretto funzionamento delle funzioni di cui sopra, avendo cura di deallocare la memoria, se necessario.

Fondamenti di Informatica T-1, 2019/2020 – Modulo 2

Prova d'Esame 3A di Giovedì 13 Febbraio 2020 – tempo a disposizione 2h

Esercizio 2 – Ricerca Appartamento (moduli element.h/.c e palazzo.h/.c)

Si definisca la funzione:

```
Appartamento * trovaAppartamento(list appartamenti,
                                 int pianoMin, float mqMin, int* dim);
```

che, ricevuta in ingresso una lista di strutture dati di tipo **`Appartamento`**, cerchi gli appartamenti che si trovano ad un piano uguale o superiore al parametro **`pianoMin`**, ed abbiano una superficie uguale o superiore a **`mqMin`**. La funzione allochi la memoria minima necessaria e popoli un array di strutture di tipo **`Appartamento`**, e ne restituisca il puntatore: l'array dovrà essere riempito coi soli appartamenti che soddisfano i requisiti richiesti. Inoltre, la funzione restituisca il numero di appartamenti effettivamente trovati tramite il parametro **`dim`** (puntatore a int) passato in ingresso. L'array di appartamenti restituito deve essere ordinato per piano decrescente ed a parità di piano per superficie decrescente. Per l'ordinamento si utilizzi l'algoritmo bubble sort.

Esercizio 3 – Registrazione offerta (moduli element.h/.c e palazzo.h/palazzo.c)

Si definisca una struttura dati **`Offerta`** per memorizzare i dati relativi ad una offerta, come sopra descritta in precedenza.

Si definisca la funzione:

```
Offerta leggiUnOfferta(FILE * fp);
```

che, ricevuto in ingresso un puntatore ad una struttura dati di tipo **`FILE`**, facente riferimento ad un file con le offerte, legga i dati contenuti su una sola riga e li restituisca tramite una struttura dati di tipo **`Offerta`**. Qualora non sia possibile leggere, la funzione restituisca un offerta con identificativo dell'unità immobiliare pari a -1.

Si sviluppi poi una funzione:

```
int registraOfferta(int idAppartamento, char* cliente, float importo);
```

che, ricevuti in ingresso l'identificatore di un appartamento, il nome e cognome del cliente in un'unica stringa, l'offerta in €, proceda nel seguente modo. Accedendo al file "offerte.txt":

- Se nel file c'è già un'offerta per lo stesso appartamento da parte della stessa persona (a prescindere dal valore delle due offerte), si stampi un messaggio di errore; la funzione restituisca il valore -3;
- Se nel file c'è già un'offerta per lo stesso appartamento con valore più alto, allora si stampi un messaggio che indica che l'offerta è troppo bassa; la funzione restituisca il valore -5;
- Se nel file non c'è nessuna offerta per quell'appartamento o tutte le offerte per quell'appartamento sono più basse, allora inserisca nel file una nuova riga con l'offerta ricevuta; la funzione restituisca il valore 0.

Esercizio 4 -- Stampa dei risultati, e de-allocazione memoria (main.c)

Il candidato realizzi nella funzione **`main(…)`** un programma che chieda all'utente di specificare un piano ed una superficie in metri quadri e stampi a video tutti gli appartamenti che siano almeno a quel piano e abbiano almeno quella superficie in metri quadri. Il programma chieda poi all'utente di specificare un id di appartamento, il suo nome e cognome ed un'offerta in €: se possibile, il programma aggiunga l'offerta al file "offerte.txt".

Al termine del programma, il candidato abbia cura di de-allocare tutta la memoria allocata dinamicamente, ivi compresa la memoria allocata per le liste.

Fondamenti di Informatica T-1, 2019/2020 – Modulo 2

Prova d'Esame 3A di Giovedì 13 Febbraio 2020 – tempo a disposizione 2h

"element.h":

```
#pragma once
#define _CRT_SECURE_NO_WARNINGS

#ifndef _ELEMENT_H
#define _ELEMENT_H

typedef struct Appartamento {
	int id;
	char natura;
	float mq;
	int piano;
	int vani;
	float mqVani[5];
} Appartamento;

typedef Appartamento element;

#define DIM_NOME 2048
typedef struct {
	int id;
	char nome[DIM_NOME];
	float valore;
} Offerta;

int compare(Appartamento a1, Appartamento a2);

#endif
```

"element.c":

```
#pragma once
#include "element.h"
int compare(Appartamento a1, Appartamento a2) {
	int result;

	result = (a2.piano - a1.piano);
	if (result == 0) {
		if (a1.mq > a2.mq)
			result = -1;
		else
			if (a1.mq < a2.mq)
				result = 1;
			else
				result = 0;
	}

	return result;
}
```

Fondamenti di Informatica T-1, 2019/2020 – Modulo 2

Prova d'Esame 3A di Giovedì 13 Febbraio 2020 – tempo a disposizione 2h

"list.h":

```
#pragma once
#ifndef LIST_H
#define LIST_H

#include "element.h"

typedef struct list_element {
	element value;
	struct list_element *next;
} item;

typedef item* list;

/* PRIMITIVE  */
list emptylist(void);
int empty(list);
list cons(element, list);
element head(list);
list tail(list);

void showlist(list l);
void freelist(list l);

#endif
```

"list.c:"

```
#include "list.h"

/* OPERAZIONI PRIMITIVE */
list emptylist(void) {		/* costruttore lista vuota */
	return NULL;
}

int empty(list l) { /* verifica se lista vuota */
	return (l == NULL);
}

list cons(element e, list l) {
	list t;		/* costruttore che aggiunge in testa alla lista */
	t = (list)malloc(sizeof(item));
	t->value = e;
	t->next = l;
	return(t);
}

element head(list l) { /* selettore testa lista */
	if (empty(l)) exit(-2);
	else return (l->value);
}

list tail(list l) {		/* selettore coda lista */
	if (empty(l)) exit(-1);
	else return (l->next);
}

void showlist(list l) {
	element temp;
	if (!empty(l)) {
		temp = head(l);
		printf("%d %c %d %f %d\n",
			temp.id, temp.natura, temp.piano, temp.mq, temp.vani);
```

```
                showlist(tail(l));
                return;
        }
        else {
                printf("\n\n");
                return;
        }
}

void freelist(list l) {
        if (empty(l))
                return;
        else {
                freelist(tail(l));
                free(l);
        }
        return;
}
```

Fondamenti di Informatica T-1, 2019/2020 – Modulo 2

Prova d'Esame 3A di Giovedì 13 Febbraio 2020 – tempo a disposizione 2h

"palazzo.h":

```
#define _CRT_SECURE_NO_WARNINGS

#pragma once
#include <stdio.h>
#include "element.h"
#include "list.h"

// Es. 1
Appartamento leggiUnAppartamento(FILE * fp);
list leggiTuttiAppartamenti(char * fileName);

// Es. 2
Appartamento* trovaAppartamento(list appartamenti, int pianoMin, float mqMin, int* dim);

// Es. 3
Offerta leggiUnOfferta(FILE * fp);
int registraOfferta(char * filename, int idAppartamento, char* cliente, float offerta);
```

"palazzo.c":

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdlib.h>
#include <string.h>
#include "palazzo.h"

Appartamento leggiUnAppartamento(FILE * fp) {
	Appartamento a;
	int i;
	if (fscanf(fp, "%d %c %f %d %d", &a.id, &a.natura, &a.mq, &a.piano, &a.vani) == 5)
{
		for (i = 0; i < a.vani; i++) {
			fscanf(fp, "%f", &a.mqVani[i]);
		}
	}
	else
		a.id = -1;
	return a;
}

list leggiTuttiAppartamenti(char * fileName) {
	FILE * fp;
	list l = emptylist();
	Appartamento temp;
	if ((fp = fopen(fileName, "rt")) != NULL) {
		while ((temp = leggiUnAppartamento(fp)).id != -1)
			l = cons(temp, l);
		fclose(fp);
	}
	return l;
}
void scambia(Appartamento* e1, Appartamento* e2) {
	Appartamento tmp = *e1;
	*e1 = *e2;
	*e2 = tmp;
}
void bubbleSort(Appartamento v[], int n) {
	int i, ordinato = 0;
	while (n > 1 && !ordinato) {
		ordinato = 1;
		for (i = 0; i < n - 1; i++)
			if (compare(v[i], v[i + 1]) > 0) {
				scambia(&v[i], &v[i + 1]);
```

```
                        ordinato = 0;
                }
            n--;
        }
}

Appartamento* trovaAppartamento(list appartamenti, int pianoMin, float mqMin, int* dim) {
        list l = appartamenti;
        Appartamento temp;
        Appartamento* result;
        int i;
        *dim = 0;
        while (!empty(l)) {
            temp = head(l);
            if (temp.piano >= pianoMin && temp.mq >= mqMin) {
                *dim = *dim + 1;
            }
            l = tail(l);
        }

        result = (Appartamento *)malloc(sizeof(Appartamento)*(*dim));
        l = appartamenti;
        i = 0;
        while (!empty(l)) {
            temp = head(l);
            if (temp.piano >= pianoMin && temp.mq >= mqMin) {
                result[i] = temp;
                i++;
            }
            l = tail(l);
        }
        bubbleSort(result, *dim);
        return result;
}

// Es. 3
Offerta leggiUnOfferta(FILE * fp) {
        Offerta temp;
        int i;
        char ch;

        if (fscanf(fp, "%d", &(temp.id)) == 1) {
            // ciclo di scarto dei caratteri davanti al nome
            do {
                ch = fgetc(fp);
            } while (ch != '"');

            i = 0;
            ch = fgetc(fp);
            while (ch != '"' && i < DIM_NOME - 1) {
                temp.nome[i] = ch;
                i++;
                ch = fgetc(fp);
            }
            temp.nome[i] = '\0';

            fscanf(fp, "%f", &(temp.valore));
        }
        else {
            temp.id = -1;
        }
        return temp;
}

int registraOfferta(char * filename, int offId, char* offCli, float offVal) {
```

Fondamenti di Informatica T-1, 2019/2020 – Modulo 2

Prova d'Esame 3A di Giovedì 13 Febbraio 2020 – tempo a disposizione 2h

```
	FILE * fp;
	Offerta prev;
	int stessoCliente = 0;
	int offertaPiuAlta = 0;
	int result = 0;
	if ((fp = fopen(filename, "rt")) != NULL) {
		// col cilo seguente determino, scorrendo tutto il file,
		// quali dei tre casi valgono...
		prev = leggiUnOfferta(fp);
		while (!stessoCliente && prev.id != -1 ) {
			// caso 1
			if ( strcmp(offCli, prev.nome) == 0 && offId == prev.id ) {
				stessoCliente = 1;
			}
			// caso 2
			if (prev.id == offId && prev.valore >= offVal) {
				offertaPiuAlta = 1;
			}
			if (!stessoCliente)
				prev = leggiUnOfferta(fp);
		}
		fclose(fp);
		// caso 1:
		if (stessoCliente) {
			printf("ERRORE: %s ha gia' fatto un'offerta per l'app %d.\n", offCli,
offId);
			result = -3;
		}
		// caso 2:
		if (!stessoCliente && offertaPiuAlta) {
			printf("Esiste almeno un'offerta piu' alta...\n");
			result = -5;
		}
		// caso 3:
		if (!stessoCliente && !offertaPiuAlta) {
			if ((fp = fopen(filename, "at")) != NULL) {
				fprintf(fp, "%d \"%s\" %f\n", offId, offCli, offVal);
				fclose(fp);
			}
			result = 0;
		}
	}
	return result;
}
```

Fondamenti di Informatica T-1, 2019/2020 – Modulo 2

Prova d'Esame 3A di Giovedì 13 Febbraio 2020 – tempo a disposizione 2h

"main.c":

```
#define _CRT_SECURE_NO_WARNINGS

#include <stdio.h>
#include <stdlib.h>
#include "palazzo.h"

int main() {
	list l;
	int dim, i;
	Appartamento* buoni;
	int pianoCercato;
	float superficieCercata;
	int idApp;
	char nomeCliente[DIM_NOME];
	float valoreOfferta;
	char ch;

	l = leggiTuttiAppartamenti("palazzo.txt");
	showlist(l);

	printf("Inserire piano e superficie cercata: ");
	scanf("%d%f", &pianoCercato, &superficieCercata);
	// buoni = trovaAppartamento(l,3,80,&dim);
	buoni = trovaAppartamento(l, pianoCercato, superficieCercata, &dim);
	for ( i = 0; i < dim; i++) {
		printf("%d %c %d %f %d\n",
			buoni[i].id,	buoni[i].natura,	buoni[i].piano,	buoni[i].mq,
buoni[i].vani);
	}

	// questa deve restituire "offerta già presente dallo stesso cliente"
	registraOfferta("offerte.txt", 1, "Piero Flacco", 100000.0);
	// questa deve restituire "offerta troppo bassa"
	registraOfferta("offerte.txt", 4, "Piero Flacco", 150000.0);
	// questa deve andare a buon fine
	registraOfferta("offerte.txt", 1, "Federico Chesani", 114000.0);


	printf("Specificare	id,	nome	cliente	(tra	doppie	virgolette)	e	valore
dell'offerta:\n");
	scanf("%d", &idApp);
	do {
		ch = getchar();
	} while (ch != '"');
	i = 0;
	ch = getchar();
	while (ch != '"' && i < DIM_NOME - 1) {
		nomeCliente[i] = ch;
		i++;
		ch = getchar();
	}
	nomeCliente[i] = '\0';
	scanf("%f", &(valoreOfferta));
	registraOfferta("offerte.txt", idApp, nomeCliente, valoreOfferta);

	freelist(l);
	free(buoni);

	return 0;
}
```

Fondamenti di Informatica T-1, 2019/2020 – Modulo 2

Prova d'Esame 3A di Giovedì 13 Febbraio 2020 – tempo a disposizione 2h

"palazzo.txt":

```
1 C 101.5 1 5 40.5 20 12.5 18.5 10
2 C 55.5 1 3 20.5 10.5 25
3 C 120.5 2 5 20.5 40 18.5 12.5 10
4 C 45.5 2 4 20 9 8 8.5
5 C 70.5 3 5 20.5 20 12.5 8.5 6
6 C 80 3 5 30.5 20 12.5 8.5 10
7 C 60.5 4 2 40 20.5
8 C 110.5 4 3 50 30.5 20
```

"offerte.txt":

```
1 "Piero Flacco" 100000.00
2 "Federico Chesani" 110000.00
2 "Piero Flacco" 150000.000
4 "Paola Mello" 250000.000
```